# 第八周作业

## BlockBasedMatching

## 沈宗毅2015141461015

1. 请解释以下变量：【维度、含义】

Demo.m -> opts.BlockSize (Line 22)

块匹配中快的大小，这里用边长占得像素点个数表示。维度是1x1的整数类型

Demo.m -> opts.SearchLimit (Line 23)

搜索框口的大小，这里也用像素点个数表示窗口大小。维度是1x1的整数类型

Motion\_Est -> M (Line 50)

这里M是原图的高除块长度（就是之前的BlockSize），向下取整之后再乘以数据块的长度。这个是为了保证搜索的时候不超出原图片大小范围，而且尽量所有地方都能搜索到。维度是1x1的整数类型。

Motion\_Est -> N (Line 51)

这里N是原图的宽除块长度（就是之前的BlockSize），向下取整之后再乘以数据块的长度。这个是为了保证搜索的时候不超出原图片大小范围，而且尽量所有地方都能搜索到。维度是1x1的整数类型。和上面的M差不多。

Motion\_Est -> img\_ref (Line 61)

这里的这个img\_ref，是参考图像做了两个加长(padarray)操作之后的图像。首先，第一次加长是在图片四周都增加了数据块边长的一半，所以图像总体长和宽都增加了数据块边长的一倍，也就是多了一个BlockSize。第一次加长，所有增加出来的像素点都用原图中最接近它的像素点填充。然后，第二次加长是在图片四周都增加了搜索窗口的一倍，所以长和宽都增加了两倍的SerchLimit。这次所有新增加的像素点都用0填充。第一次加长是为了囊括所有有意义的点，并且给可能的整个图形平移预留空间。第二次加长纯粹是为了搜索不越界，下面的main loop主循环更纯粹一点，正如注释里的’to fit SearchLimit’。维度，假设原参考图长和宽分为为n和m,那么这里的img\_ref就是一个(n+BlockSize+2\*SearchLimit)\*（m+BlockSize+2\*SearchLimit）\*3的图片，代表了加工过的参考图片。

Motion\_Est -> img\_test (Line 62)

这个其实和上面的img\_ref经过了相同的操作。不过这个是代表我们现在的图像而不是参考图，参考图像其实就是下一帧的图片。同样这个img\_test的维度也是(n+BlockSize+2\*SearchLimit)\*（m+BlockSize+2\*SearchLimit）\*3，如果原来图片是n\*m\*3的话。

Motion\_Est -> xc\_range (Line 69)

这儿的xc\_range是每一个数据块的起始点的x坐标。注意，因为这里的img\_ref和img\_test都进行了2次加工，所以第一个数据块的起点并不是（0,0），而是BlockSize/2+SearchLimit+1，同理，最后一个数据块的终点也不是N-BlockSize，而是N-SearchLimit-BlockSize/2。维度是一个1\*(N\_i/BlockSize)的矩阵，注意这里的N\_i表示原来没加工过的图片的高。

Motion\_Est -> yc\_range (Line 70)

这个和上面的xc\_range差不多，但是是每一个数据块的起点的y坐标。同样，起点也不是(0,0)，终点也不是(M,0)，而是经过变换的BlockSize/2+SearchLimit+1和N-SearchLimit-BlockSize/2。所以可以看出来，第一个数据块的起点是(BlockSize/2+SearchLimit+1, BlockSize/2+SearchLimit+1),最后一个数据块的终点是（N-SearchLimit-BlockSize/2，M-SearchLimit-BlockSize/2）。同样，这个东西的维度是一个1\*(M/BlockSize)的矩阵。

1. 请解释以下函数：【输入、输出、功能】

[MVx MVy] = Motion\_Est(img\_test, img\_ref, opts)

这个函数的功能是返回所有数据块的光流。这个光流我的理解有点像之前说的梯度，就是变化趋势，但是这里不是颜色上的变化趋势，而是数据块在图像中位置的变化趋势。每一个数据块用一个二维向量来来描述它，这个二维向量被拆成了MVx和MVy。MVx表示这个每个数据块在x方向上的位移，MVy表示每个数据块在y方向上的位移。一共有floor(M/BlockSize)\*floor(N/BlockSize)个数据块，所以MVx和MVy都是floor(M/BlockSize)\*floor(N/BlockSize)\*1的矩阵。

[MVy MVx] = FullSearch(Block, img\_ref, xc, yc, SearchLimit)

这个函数是返回当前数据块的“光流”的，也就是这个数据块的移动方向。用穷举方法搜索出最接近当前数据块的数据块，然后做位置上的差运算，返回MVx和MVy。这里面的相似性度量用的是差的绝对值之和除以边长的一半，也就是PPT里的MAD。

g = reconstruct(img0, MVx, MVy)

这个就是根据已经有的图片和已经知道的所有光流，来拟合出下一帧。

1. 请解释输出结果：

imgM

事实上这个东西在程序里写成了imgMC,这个就是我们根据光流和这一帧的图片拟合出来的下一帧的图片。

imgM – img1

这个就是拟合出来的图片减去真实的图片的差。当然里面进行了一些容错处理，对于边界的。这个矩阵表示的是拟合出来的图片和真实图片的灰度值的差，表示了拟合的质量。

PSNR

这个值原名叫Peak Signal to Noise Ratio，峰值信噪比。首先，要知道MSE，MSE就是我们拟合出来的图像和原来图像的不相似度，主要用灰度值的平方再除以像素点个数来表示。然后，这个PSNR把MSE换了一种方式来表达，用原来真实图像MC的最大值的平方除以MSE，得到的一个没有量纲的数值，并且用分贝作为单位来表达这个数值。别看只有30不到，但其实30分贝表达的是1000倍，所以可以看出信噪比还是很高的，也就是说错误相比起原来的数值还是比较小的。

1. 修改程序，输入salesman001.png, salesman002.png，使用mean square error作为相似性度量，其PSRN值是多少？（贴上修改的代码）

PSNR是27.2065。

只需要把FullSearch.m里的Main Loop部分修改为

for i = -SearchRange:SearchRange

for j = -SearchRange:SearchRange

xt = xc + j;

yt = yc + i;

Block\_ref = img\_ref(yt+BlockRange, xt+BlockRange, :);

%SAD = sum(abs(Block(:) - Block\_ref(:)))/(BlockSize^2);

SAD = sum((Block(:) - Block\_ref(:)).^2)/(BlockSize^2);

if SAD < SADmin

SADmin = SAD;

x\_min = xt;

y\_min = yt;

end

% Motion Vector (integer part)

MVx\_int = xc - x\_min;

MVy\_int = yc - y\_min;

end

end

其实就改了一行，就是SAD那儿。我没有再规范地起个名字，为了方便起见直接沿用了SAD。程序我附在压缩包里。

1. 修改程序，输入city001.png, city002.png，计算从img1合成得到的与img0相同视角的imgM，对应的PSNR值是多少？（贴上修改的代码）

这一句话有点小小的拗口啊，我的理解是img1是这一帧，img0是下一帧。我不确定我理解对了，但我猜想city001肯定是在前的，city002肯定是在后的。而且，相似性度量我两种也都采用了。

采用MAD作为相似性度量：PSNR为29.0428

采用MSE作为相似性度量：PSNR为29.2121